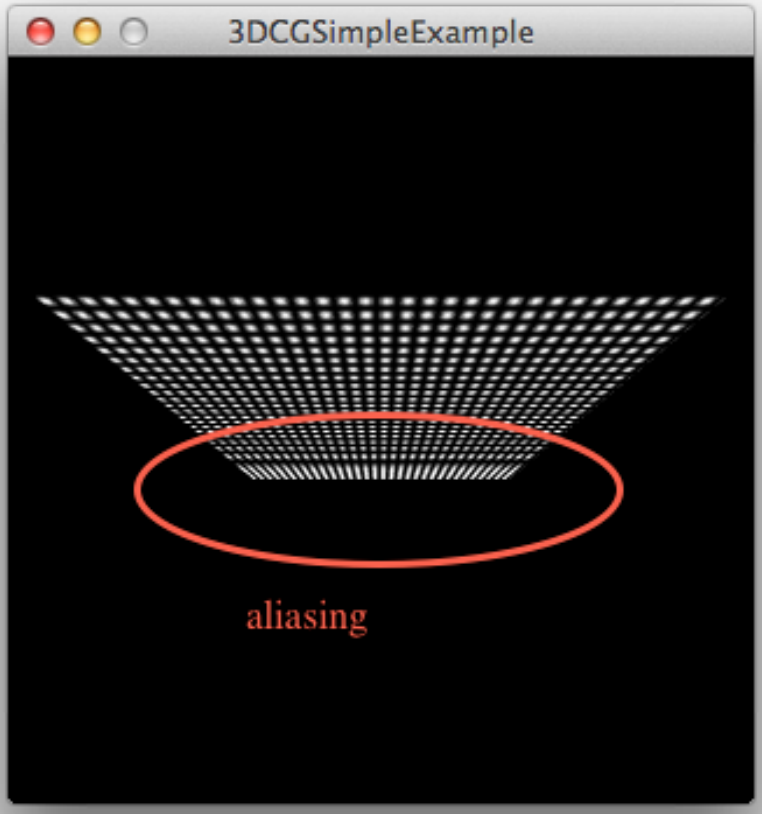
問題 1 ：Texturing

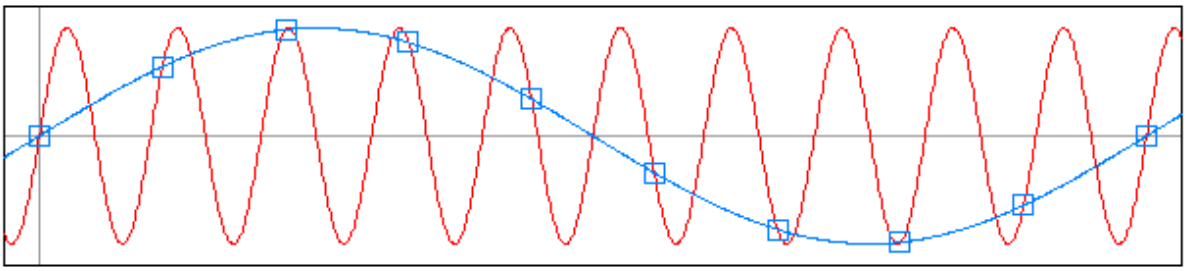
1. Explain **aliasing** artifact of texturing from the point of view of the sampling theorem. This explanation must have two parts. The first one is a description of the theoretical reason behind of the aliasing with equations and illustrations. The second one is an experimental description base on the actual artifacts occurring while the animation by the program, using snapshots of the animation.
2. Explain one texturing algorithm to avoid the artifact.

Sourcecode：**CG_Final_Code/MipMapping**

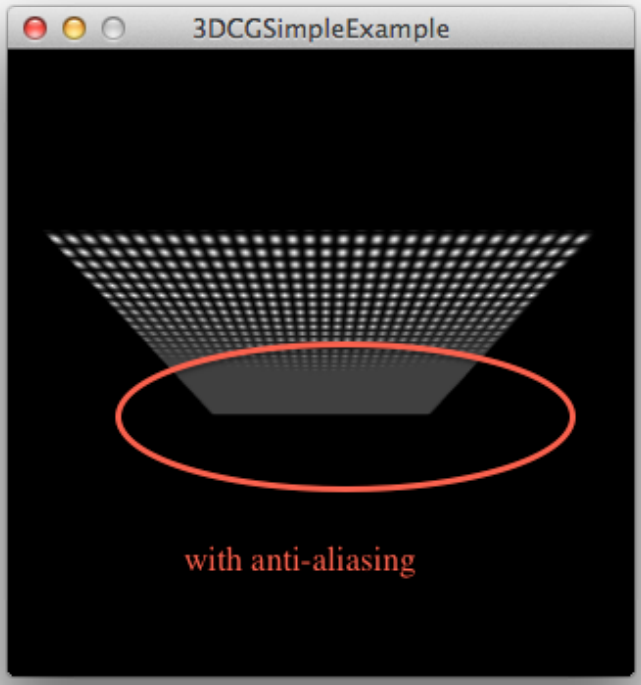
1. **aliasing**（エイリアシング）とは，信号をサンプリングを行う際，入力信号に含まれる周波数成分がサンプリング周波数の $\frac{1}{2}$ よりも高い場合に生じる折り返し歪みの現象である．例えば，我々がレンガの壁などの模様が複雑な物体をピクセルの少ない画像にするときに，モアレ（干渉縞）が生じる現象も aliasing の一種である．今回のソースコードでは，512×512 の **DotImage** を 320×320 以下にする時は，次の画像のような aliasing 現象が見られる．



2. aliasing が生じる原因は，信号の高周波成分に対し，サンプリング周波数がそのナイキスト条件（サンプリング周波数 f_s と入力周波数 f_{input} の関係は $f_s > 2f_{\text{input}}$）に満たさないこと．その場合，サンプリングを用いて高周波成分を再生する時に低周波の aliasing noise になってしまう．例えば，次の画像のように，周波数が違う赤い正弦曲線と青い正弦曲線はサンプリング化によって全く同じ標本列を生成した．そのサンプリングは青い正弦曲線を再生できるが，赤い正弦曲線を再生できない．
   aliasing を防ぐため，anti-aliasing 技術がある．一般的には，サンプリングする前に，低周波成分をそのままで，高周波成分を低周波成分に近似し，低周波として再生される．



3. CG における，代表的な anti-aliasing アルゴリズムは Mipmapping である．Mipmap とは，3DCG において，テクスチャ画像を補完する前に最適化された画像群である．この手法は描写速度を向上するだけではなく，aliasing 現象も防げる．
   Mipmapping のアルゴリズムは簡単である．メインのテクスチャの画像をあらかじめ処理し，複数のサイズに圧縮する．視点が近く場合，メインテクスチャを用いる，一方，遠くの視点に対し，圧縮された Mipmap を使う．例えば，512×512 のメインテクスチャには，サイズがそれぞれ $256 \times 256, 128 \times 128, 64 \times 64 \ldots$ の画像群を生成し，レンダリング領域の大きさによって近似する Mipmap を使えばよい，例としてレンダリング領域が 300×300 の場合，256×256 と 128×128 の二枚の Mipmap の 300×300 となる線形結合を構造できる．
   また，Mipmapping を使うと，テクスチャのサンプリングは視点の位置に関わらず必ずナイキスト条件を満たすため，aliasing に対する有効な解決手段と考えられる．
   しかし，生成された Mipmap を保存するため，元のテクスチャの四倍の記憶領域が必要となるデメリットがある．

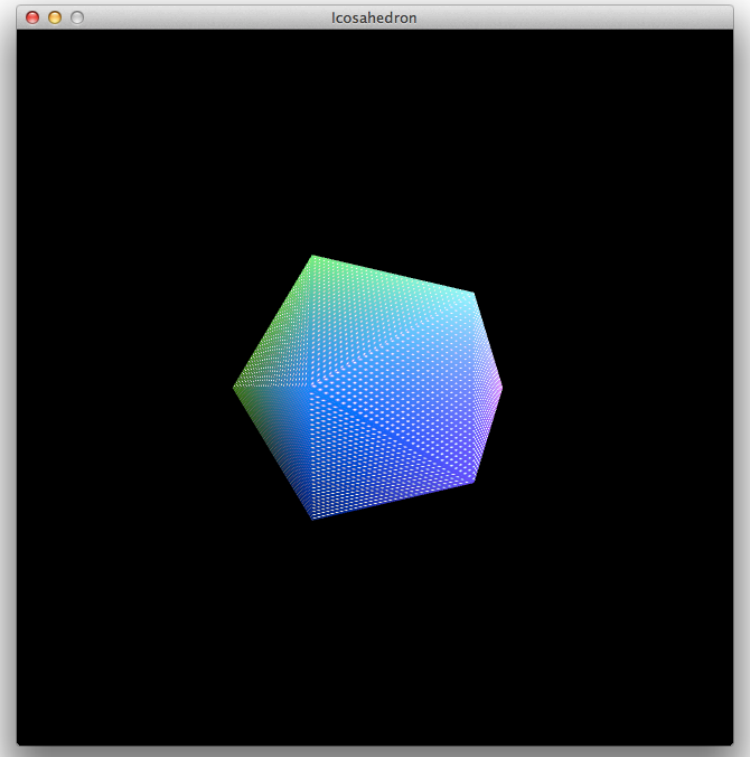


一般的なテクスチャに対するも効果がある.

問題 2 ：Geometry Data Definition and Transform

Define a regular icosahedron whose center is the origin and whose one edge length is 0.8, translate the center to $(0,\ 0,\ -2)$, and make an animation that it rotates around the vertical axis which passes through the center.
Use the same framework of the program and replace the plane to the icosahedron.
Put one snapshot on your report and describe how to compile and execute your program.

Sourcecode：**CG_Final_Code/IcosahedronNormal**（Window size を二倍に拡大した）
Screenshot:



Wikipedia により，辺の長さが a の二十面体の 12 個の頂点の座標は

$$\frac{(0, \pm 1, \pm\phi)}{2}a, \frac{(\pm 1, \pm\phi, 0)}{2}a, \frac{(\pm\phi, 0, \pm 1)}{2}a$$

ただし，$\phi = \frac{1+\sqrt{5}}{2}$．また，二十面体のすべての面は面積が $\frac{\sqrt{3}}{4}a^2$ となる正三角形である．例の **Plane** の頂点と面の配置を上記に修正すれば二十面体になる．

問題 3 ：Reflection

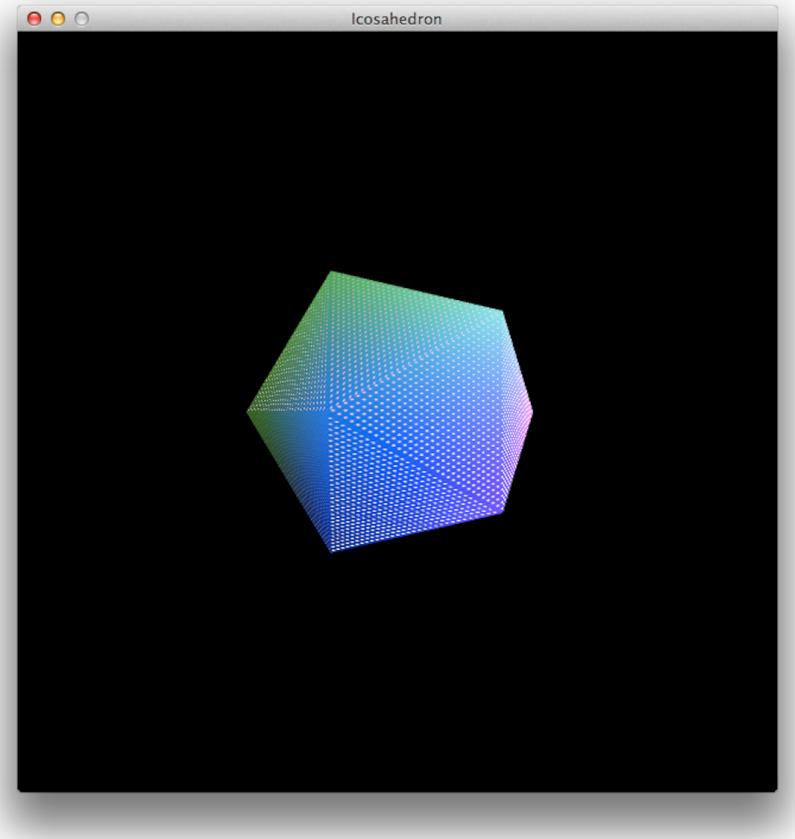
Calculate diffuse and ambient intensities on the surface of the regular icosahedron, as if it was a sphere, in the fragment shader stage. The equation to calculate the intensities R_{rgb} is defined as

$$R_{\mathrm{rgb}} = C_{\mathrm{rgb}}\Big(\max(-\frac{LN}{|L||N|}) + a\Big)$$

where C_{rgb} is the color of the icosahedron, N is the normal vector at a point, L is the vector of an incident light direction and a is a coefficient of ambient light.
Set L as an efficient direction to express shading effect on the object surface.
Put one snapshot on your report and describe how to compile and execute your program.

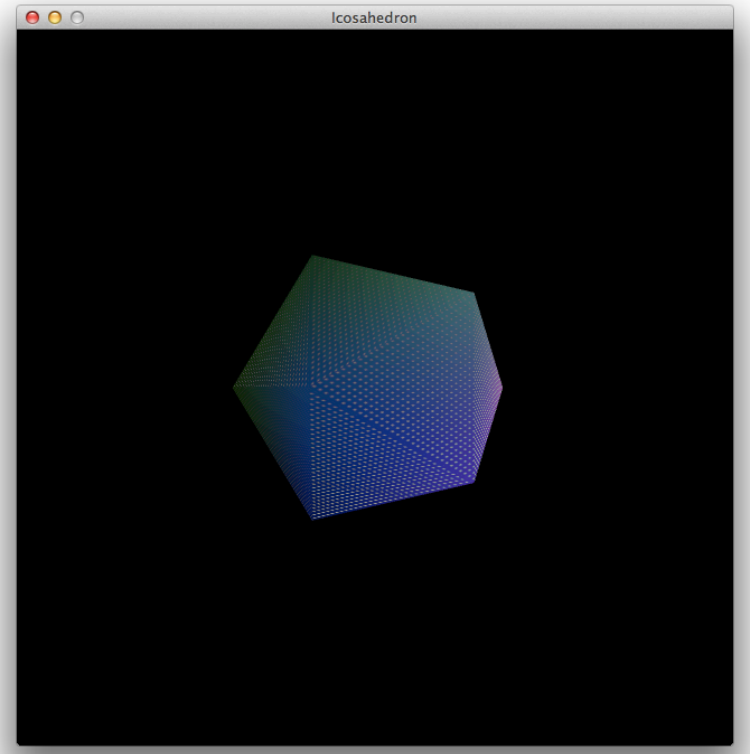
Sourcecode：**CG_Final_Code/IcosahedronWithReflection**（Window size を二倍に拡大した）

Screenshot:

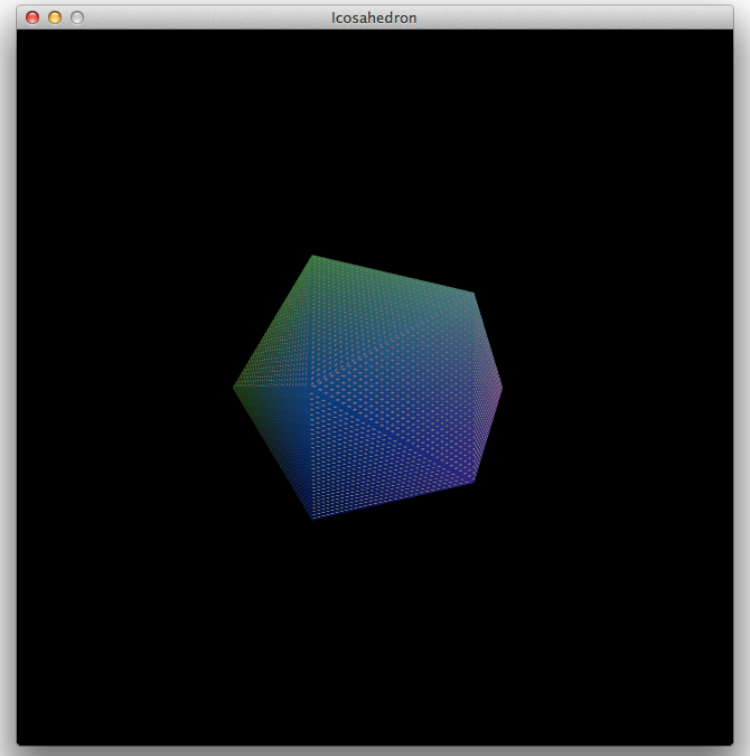


スクリーンショットで示すのは，鏡面反射（光源 $L = (0.0, 10.0, -10.0)$）と拡散（環境光係数 $a = 0.5$）の組み合せです．

次の二つの画像に分解できる．



鏡面反射



拡散